INTERNET-DRAFT There are 10 issues highlighted like this.
<draft-ietf-ipp-notify-get-~~delivery ~~00.txt>

Carl-Uno Manros
Tom Hastings
Xerox Corp.
February 3, 2000~~December 7, 1999~~

Internet Printing Protocol/1.1: **The 'ipp-notify-get' Notification Delivery Method**

ISSUE 01 - What should the name of this delivery method and protocol be that we use in the title of this document?

ISSUE 02 - What should the scheme name be? Consider 'ipp-notify-get' a working title, until we see several schemes. The 'ipp-notify-poll', 'ipp-notify-sent', and 'ipp-snmp' delivery methods are our other examples. The IETF likes words or well-recognized acronyms, not abbreviations in scheme names, so lets include "notify"?

ISSUE 03 - Should the scheme name be used in the title?

Status of this Memo

Abstract

The IPP notification specification [ipp-ntfy] is an OPTIONAL extension to IPP/1.0 and IPP/1.1 that requires the definition of one or more delivery methods for dispatching event notification reports to Notification Recipients. This document describes the semantics and syntax of the '~~ipp-get~~ipp-notify-get' event notification delivery method. For this delivery method, the client uses an explicit IPP Get-Notifications Printer operation in order to request (pull) event Notifications from the IPP Printer. The Get-Notifications request indicates whether the client wants to receive all future events Notifications for (1) any Subscription for which the client is the owner or (2) a particular Subscription object. In either case, the event Notifications are returned as MIME multi-part-related responses to the Get-Notifications request. The HTTP channel is kept open, so that subsequence event Notifications are returned using additional MIME multi-part-related responses.

The full set of IPP documents includes:

Design Goals for an Internet Printing Protocol [RFC2567]
Rationale for the Structure and Model and Protocol for the Internet Printing Protocol [RFC2568]
Internet Printing Protocol/1.1: Model and Semantics [ipp-mod]
Internet Printing Protocol/1.1: Encoding and Transport [ipp-pro]
Internet Printing Protocol/1.1: Implementer's Guide [ipp-iig]
Mapping between LPD and IPP Protocols [RFC2569]
Internet Printing Protocol/1.0 & 1.1: Event Notification Specification [ipp-ntfy]

The "Design Goals for an Internet Printing Protocol" document takes a broad look at distributed printing functionality, and it enumerates real-life scenarios that help to clarify the features that need to be included in a printing protocol for the Internet. It identifies requirements for three types of users: end users, operators, and administrators. It calls out a subset of end user requirements that are satisfied in IPP/1.0. A few OPTIONAL operator operations have been added to IPP/1.1.

The "Rationale for the Structure and Model and Protocol for the Internet Printing Protocol" document describes IPP from a high level view, defines a roadmap for the various documents that form the suite of IPP specification documents, and gives background and rationale for the IETF working group's major decisions.

The "Internet Printing Protocol/1.1: Model and Semantics" document describes a simplified model with abstract objects, their attributes, and their operations that are independent of encoding and transport. It introduces a Printer and a Job object. The Job object optionally supports multiple documents per Job. It also addresses security, internationalization, and directory issues.

The "Internet Printing Protocol/1.1: Encoding and Transport" document is a formal mapping of the abstract operations and attributes defined in the model document onto HTTP/1.1 [RFC2616]. It defines the encoding rules for a new Internet MIME media type called "application/ipp". This document also defines the rules for transporting over HTTP a message body whose Content-Type is "application/ipp". This document defines a new scheme named 'ipp' for identifying IPP printers and jobs.

The "Internet Printing Protocol/1.1: Implementer's Guide" document gives insight and advice to implementers of IPP clients and IPP objects. It is intended to help them understand IPP/1.1 and some of the considerations that may assist them in the design of their client and/or IPP object implementations. For example, a typical order of processing requests is given, including error checking. Motivation for some of the specification decisions is also included.

The "Mapping between LPD and IPP Protocols" document gives some advice to implementers of gateways between IPP and LPD (Line Printer Daemon) implementations.

The "Event Notification Specification" document defines OPTIONAL operations that allow a client to subscribe to printing related events. Subscriptions include "Per-Job subscriptions" and "Per-Printer subscriptions". Subscriptions are modeled as Subscription objects. Four other operations are defined for subscription objects: get attributes, get subscriptions, renew a subscription, and cancel a subscription.

Table of Contents

1 Introduction 4
2 Terminology 4
3 Model and Operation 4
4 Get-Notifications operation 5
4.1 GET-NOTIFICATIONS REQUEST 6
4.2 GET-NOTIFICATIONS RESPONSE 7
5 Encoding 7
6 IANA Considerations 8
7 Internationalization Considerations 8
8 Security Considerations 8
9 References 9
10 Author's Addresses 9
11 Full Copyright Statement 10

1 Introduction

IPP printers that support the OPTIONAL IPP notification extension [ipp-ntfy] either a) accept, store, and use notification subscriptions to generate notification reports and implement one or more delivery methods for notifying interested parties, or b) support a subset of these tasks and farm out the remaining tasks to a Notification Delivery Service. The '~~ipp-get~~ipp-notify-get' event notification delivery method specified in this document defines a Get-Notifications operation that may be used in a variety of notification scenarios. Its primary intended use is for clients that want to be Notification Recipients to explicitly request (pull) event Notifications from the IPP Printer upon request. However, the Get-Notifications operation may also be used by Notification Delivery Services to request (pull) event Notifications from an IPP Printer for subsequent distribution to the Ultimate Notification Recipients. The HTTP channel is kept open, so that subsequence event Notifications are returned using additional MIME multi-part-related responses.

2 Terminology

This document uses terms such as "attributes", "keywords", and "support". These terms have special meaning and are defined in the model terminology [ipp-mod] section 12.2.

Capitalized terms, such as MUST, MUST NOT, REQUIRED, SHOULD, SHOULD NOT, MAY, NEED NOT, and OPTIONAL, have special meaning relating to conformance. These terms are defined in [ipp-mod] section 12.1 on conformance terminology, most of which is taken from RFC 2119 [RFC2119].

This section defines the following additional terms that are used throughout this document:

REQUIRED: if an implementation supports the extensions described in this document, it MUST support a REQUIRED feature.
OPTIONAL: if an implementation supports the extensions described in this document, it MAY support an OPTIONAL feature.
Notification Recipient - See [ipp-ntfy]
Subscription object - See [ipp-ntfy]
Ultimate Notification Recipient - See [ipp-ntfy]

3 Model and Operation

In the IPP Notification Model [ipp-ntfy], one or more Per-Job Subscriptions can be supplied in the Job Creation operation or OPTIONALLY as subsequent Create-Job-Subscription operations; one Per-Printer Subscription can be supplied in the Create-Printer operation. The client that creates these Subscription objects becomes the owner of the Subscription object.

When creating each Subscription object, the client supplies the "notify-recipient" (uri) attribute. The "notify-recipient" attribute specifies both a single Notification Recipient that is to receive the event Notifications when subsequent events occur and the method for notification delivery that the IPP Printer is to use. For the Notification delivery method defined in this document, the notification method is '~~ipp-get~~ipp-notify-get', and the Notification Recipient is omitted, since any client that is authenticated (1) as an operator or administrator or (2) as the owner of the Subscription object can initiate a Get-Notifications operation for that Subscription object. Thus a single user can login at different places, say his/her office, the lab, and/or several desktops in the same room, and receive the same event Notifications from a single Subscription object.

For the '~~ipp-get~~ipp-notify-get' event notification delivery method defined in the document, the client who created the Subscription objects is also the Notification Recipient. The client issues a Get-Notifications Printer operation in order to initiate the delivery of the next event Notifications that occur. The client can indicate in the Get-Notifications request whether it wants to receive all future event Notifications for (1) any existing or future Subscription objects for which it is the owner or (2) a particular Subscription object (for which it MUST be the owner). In either case, the Notifications are returned as MIME multi-part-related responses to the Get-Notifications request. The HTTP channel is kept open for an indefinite period, so that the IPP Printer continues to return additional parts of the MIME multi-part-related responses for each event Notification as it occurs. Either the client or the IPP Printer can disconnect the HTTP connection. However, if the IPP Printer grants an HTTP connection it SHOULD disconnect only under unusual circumstances.

ISSUE 04: Is there a limit to the number of outstanding Get-Notifications requests that an IPP Printer supports? What is this number? How does it relate to the maximum number of Subscriptions? Can the client determine the number?

ISSUE 05: Should an implementation be able to queue event Notifications, so that a client can get event Notifications that had occurred prior to the Get-Notifications? If so, how long does the IPP Printer keep the event Notifications before discarding them (for this delivery method only)? The lease time of the Subscription object? If this is possible, should the subscriber get to say whether to queue or not, or is it just baked into the implementation. If the former, does the subscriber indicate via a parameter in the notification method URL? If the latter, how does a client discover whether event Notifications are queued or not? Should we have two different notification methods, one the queues and one that doesn't?

From the December meeting:

```
It was suggested that any "notification queuing service" should
be the responsibility of the Notification Recipient, not the
Printer.  However, the Issue was not completely resolved.
```

4 Get-Notifications operation

This REQUIRED operation allows the client to request that future event Notifications be delivered as MIME multi-part-related responses to this request. The client MUST be the owner of the Subscription objects that are involved and the delivery method specified when the Subscription objects were created MUST be '~~ipp-get~~ipp-notify-get'. However, the client can and SHOULD issue the Get-Notifications request before having created any Subscription objects, in order not to miss any event Notifications.

The IPP Printer MUST accept the request in any state (see [ipp-mod] "printer-state", ~~and~~ "printer-state-reasons", and "printer-is-accepting-jobs" attributes) and MUST remain in the same state with the same "printer-state-reasons".

Current "printer-state"	New "printer-state"	new "printer-state-reasons"	IPP Printer's response status code and action:
'idle'	'idle'	no change	'successful-ok'
'processing'	'processing'	no change	'successful-ok'
'stopped'	'stopped'	no change	'successful-ok'

~~ISSUE 03: What "printer-state-reasons" might cause an error return, if any? 'paused', 'shutdown', 'quiescent'?~~

Access Rights: The authenticated user (see [ipp-mod] section 8.3) performing this operation must either be the Subscription object owner (as determined when the Subscription object was created by the Job Creation operation, Create-Job-Subscription, or Create-Printer-Subscription operations) or an operator or administrator of the Printer object (see [ipp-mod] Sections 1 and 8.5). Otherwise, the IPP object MUST reject the operation and return: 'client-error-forbidden', 'client-error-not-authenticated', or 'client-error-not-authorized' as appropriate.

4.1 Get-Notifications Request

The following groups of attributes are part of the Get-Notifications Request:

Group 1: Operation Attributes

Natural Language and Character Set:
The "attributes-charset" and "attributes-natural-language" attributes as described in [ipp-mod] section 3.1.4.1.

Target:
The "printer-uri" (uri) operation attribute which is the target for this operation as described in [ipp-mod] section 3.1.5.

Requesting User Name:
The "requesting-user-name" (name(MAX)) attribute SHOULD be supplied by the client as described in [ipp-mod] section 8.3.

"subscription-id" (integer(1:MAX)):
The client OPTIONALLY supplies this attribute. The Printer object MUST support this attribute. It is an integer value that identifies the Subscription object for which event Notifications are being requested. If the client supplies this attribute, but the Subscription object is not found, the IPP Printer MUST return the 'client-error-not-found' status code. If the client does not supply this attribute, then the IPP Printer returns event Notifications for all Subscription objects for which the client is the owner and the "notify-recipients" attribute is '~~ipp-get~~ipp-notify-get'. It is not an error if there are currently no Subscription objects for this client; the client can create Subscription objects later that will start returning event Notifications as responses to this operation.

4.2 Get-Notifications Response

The Printer object returns either an immediate error response or a successful response with status code: 'successful-ok' when the first event occurs, i.e., when the Printer delivers the first event Notification.

Group 1: Operation Attributes

Status Message:
In addition to the REQUIRED status code returned in every response, the response OPTIONALLY includes a "status-message" (text(255)) and/or a "detailed-status-message" (text(MAX)) operation attribute as described in [ipp-mod] sections 13 and 3.1.6.

Natural Language and Character Set:
The "attributes-charset" and "attributes-natural-language" attributes as described in [ipp-mod] section 3.1.4.2.

Group 2: Unsupported Attributes

See [ipp-mod] section 3.1.7 for details on returning Unsupported Attributes.

Group 3: ~~Generic Object~~Notification Attributes

The Printer object responds with one event Notification (see [ipp-ntfy]). If there are multiple events that occur at the same time, the Printer object returns them in separate MIME multi-part-related responses, each as separate IPP operation responses, as well. The HTTP channel is kept open for an indefinite period, so that the IPP Printer continues to return additional parts of the MIME multi-part-related responses for each event Notification as it occurs.
ISSUE 06 - Is this correct for MIME multi-part-related responses? This need prototyping.
ISSUE 07 - What happens if 100 continue isn't supported?
ISSUE 08 - What happens if HTTP keep alive isn't supported?

5 Encoding

The operation-id assigned for the Get-Notification operation is:

0x00??

and should be added to the next version of [ipp-mod] section 4.4.15 "operations-supported".

This notification delivery method uses the IPP transport and encoding [ipp-pro] for the Get-Notifications operation with one extension:

~~Instead of defining a new object attribute tag, a Generic Object attributes tag is defined that is used for all new objects, such as Subscription objects, etc. Then this one new tag can also be used for the Get-Notifications response Group 3 tag in section 4.2:~~

~~generic-object~~notification-attributes-tag = %x07~~??~~ ; tag of 07~~?~~

ISSUE 9 - The problem with assigning new tags for every new kind of attributes and objects, is that an implementation that does private or experimental operations that have new kinds of attributes and/or objects, will be forced to either overload some existing tag or use one of the tags reserved for future standardization. See email from Ned Freed about the need to clarify [ipp-pro] about:

0x06-0x0e	reserved for future delimiters
0x0F	reserved for future chunking-end-of-attributes-tag

0x11	reserved for future 'default'
0x14-0x1F	reserved for future "out-of-band" values.

Whereas if we had a generic tag, that same tag could be used for the private and experimental operations. The Printer and the client then uses the operation-id itself to determine what kind of attributes or object is being passed in the request or returned in the response, respectively.

Another possible approach would be to assign one tag for private use and then keep assigning new tags for standard uses, such as Subscription (0x6) and Notification (0x7).

6 IANA Considerations

IANA will be asked to register this '~~ipp-get~~ipp-notify-get' notification delivery scheme.

ISSUE 10: Any notification delivery scheme has to be registered with IANA, since it is a URL scheme, correct?

7 Internationalization Considerations

With the '~~ipp-get~~ipp-notify-get' method defined in this document, the client cannot request the Human Consumable form by supplying the "notify-text-format" operation attribute (see [ipp-ntfy]). Therefore, the IPP Printer does not have to perform any localization with this notification delivery method. However, the client when it receives the Get-Notifications response is expected to localize the attributes that have the 'keyword' attribute syntax according to the charset and natural language requested in the Get-Notifications request.

8 Security Considerations

The IPP Model and Semantics document [ipp-mod] discusses high level security requirements (Client Authentication, Server Authentication and Operation Privacy). Client Authentication is the mechanism by which the client proves its identity to the server in a secure manner. Server Authentication is the mechanism by which the server proves its identity to the client in a secure manner. Operation Privacy is defined as a mechanism for protecting operations from eavesdropping.

Unlike other event Notification delivery methods in which the IPP Printer initiates the event Notification, with the method defined in this document, the Notification Recipient is the client who issues the Get-Notifications operation. Therefore, there is no chance of "spam" notifications with this method. Furthermore, such a client can close down the HTTP channel at any time, and so can avoid future unwanted event Notifications at any time.

9 References

[ipp-mod]
R. deBry, T. Hastings, R. Herriot, S. Isaacson, P. Powell, "Internet Printing Protocol/1.0: Model and Semantics", <draft-ietf-ipp-model-v11-04.txt>, June, 1999.

[ipp-ntfy]
Isaacson, S., Martin, J., deBry, R., Hastings, T., Shepherd, M., Bergman, R., "Internet Printing Protocol/1.1: IPP Event Notification Specification", <draft-ietf-ipp-not-spec-02~~1~~.txt>, ~~October 14, 1999~~February 2, 2000.

[ipp-pro]
Herriot, R., Butler, S., Moore, P., Tuner, R., "Internet Printing Protocol/1.1: Encoding and Transport", draft-ietf-ipp-protocol-v11-03.txt, June, 1999.

[rfc2026]
S. Bradner, "The Internet Standards Process -- Revision 3", RFC 2026, October 1996.

[RFC2616]
R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616, June 1999.

10 Author's Addresses

Carl-Uno Manros
Xerox Corporation
737 Hawaii St. ESAE 231
El Segundo, CA 90245

Phone: 310-333-8273
Fax: 310-333-5514
e-mail: cmanros@cp10.es.xerox.com

Tom Hastings
Xerox Corporation
737 Hawaii St. ESAE 231
El Segundo, CA 90245

Phone: 310-333-6413

Fax: 310-333-5514
e-mail: hastings@cp10.es.xerox.com

11 Change History

This section lists the changes made to the document. It does not list additions or deletions of issues.

11.1 Changes made to the December 7, 1999 version to make the February 3, 2000 version

The following changes were made to the December 7, 19999 version to make the February 3, 2000 version:

1. Changed the scheme name and title from 'ipp-get' to 'ipp-notify-get'.

2. Changed the tag delimiter from generic-attributes-tag to notification-attributes-tag as agreed at the December IPP WG meeting.

12 Full Copyright Statement